MAKALAH SISTEM POINT OF SALE (POS) UNTUK CAFE

Dosen Pengampu : Arif Rohmadi S.KOM., M.CS.

Disusun Oleh : Kelompok 11

Daffa Arkan Taqiya	L0124047
Hendrata Wibsar Mulyasaputra	L0124056
Dzaki Rizal Kurniawan	L0124096

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Di era digital saat ini, banyak toko kecil seperti warung kelontong, toko sembako atau usaha mikro lainnya masih menggunakan pencatatan manual melalui buku atau kertas. Cara ini tidak hanya memakan waktu, tetapi juga rentan terjadi kesalahan dalam perhitungan, menyebabkan ketidakakuratan stok serta menyulitkan dalam memantau aktivitas penjualan harian. Akibatnya, efisiensi operasional terganggu dan pemilik usaha kesulitan mengambil keputusan bisnis yang tepat.

Untuk mengatasi persoalan tersebut, diperlukan sistem yang sederhana, ringan dan mudah dioperasikan oleh pengguna awam. Aplikasi Point of Sale (POS) untuk Cafe hadir sebagai solusi yang dirancang khusus untuk membantu pencatatan transaksi penjualan, pengelolaan stok serta tampilan struk secara otomatis. Aplikasi ini dilengkapi fitur utama seperti manajemen produk, transaksi di kasir dengan opsi diskon, dashboard ringkasan operasional dan sistem login berbasis tiga peran yaitu kasir, manager dan admin.

Aplikasi ini dikembangkan dengan arsitektur ringan menggunakan SQLite sebagai basis datanya, sehingga dapat dijalankan di perangkat desktop biasa. Interfacenya dirancang secara intuitif dan mendukung tiga tingkat akses: Kasir, Manager dan Admin. Dengan demikian, aplikasi ini menjadi sarana praktis dan terjangkau bagi pelaku usaha mikro untuk mempercepat adopsi teknologi digital tanpa biaya besar.

1.2. Rumusan Masalah

Berdasarkan latar belakang di atas, beberapa masalah utama yang dihadapi oleh toko-toko kecil dalam pengelolaan operasionalnya adalah:

1. Pencatatan transaksi yang manual dan tidak terpusat, menyebabkan potensi kesalahan hitung, hilangnya data dan sulitnya melakukan audit.
2. Tidak adanya sistem manajemen stok yang real time, sehingga sering terjadi kelebihan atau kekurangan stok.
3. Tidak adanya ringkasan penjualan harian yang mudah dipantau, sehingga pemilik toko kesulitan mengevaluasi performa operasional.

4. Proses pembayaran dan pencetakan struk yang lambat dan tidak terintegrasi.
5. Belum ada sistem login yang membedakan peran secara jelas antara kasir, manager dan admin sehingga berpotensi menimbulkan kesalahan atau penyalahgunaan akses.

1.3. Tujuan Pembuatan Aplikasi

Tujuan utama dari pembuatan aplikasi Point of Sale (POS) untuk Cafe adalah:

1. Menciptakan sistem pencatatan transaksi penjualan yang akurat dan cepat melalui fitur kasir yang memungkinkan pemilihan produk, perhitungan otomatis, penerapan diskon dan proses pembayaran tunai.
2. Mengelola katalog produk secara terstruktur dengan fitur CRUD (Create, Read, Update, Delete) untuk data produk (kode, nama, harga, stok, kategori).
3. Memantau stok barang secara real time, dengan pengecekan stok sebelum checkout dan pencegahan stok negatif.
4. Menyediakan dashboard operasional real time bagi manajer dan admin, yang menampilkan:
    - Penjualan hari ini
    - Jumlah transaksi
    - Rata-rata nilai transaksi (average ticket)
    - Produk terlaris hari ini
    - Notifikasi stok rendah
    - Transaksi terbaru
5. Menerapkan sistem keamanan berbasis tiga peran:
    - Kasir: hanya dapat melakukan transaksi.
    - Manajer: dapat transaksi, kelola produk, dan melihat dashboard serta riwayat transaksi.
    - Admin: memiliki semua hak akses manajer, ditambah kemampuan menghapus riwayat transaksi.
6. Membangun aplikasi yang ringan, mudah digunakan dan dapat dijalankan di perangkat standar tanpa memerlukan infrastruktur server atau koneksi internet.

BAB II

SOLUSI DAN PENYELESAIAN

2.1. Deskripsi Solusi

Aplikasi POS dikembangkan sebagai solusi digital yang sederhana dan fokus pada kebutuhan operasional inti cafe. Aplikasi ini dibangun menggunakan Java dan SQLite sebagai database lokal, tanpa ketergantungan pada server atau internet.

Solusi yang ditawarkan meliputi:

1. Modul Manajemen Produk: Hanya untuk Manajer dan Admin. Memungkinkan penambahan, pengeditan dan penghapusan produk, yang langsung memengaruhi daftar di kasir.
2. Modul Kasir: Tersedia untuk Kasir, Manajer, dan Admin. Memungkinkan pemilihan produk, penambahan ke keranjang, penerapan diskon, perhitungan otomatis, pembayaran tunai dan tampilan struk digital. Stok diperbarui langsung setelah transaksi.
3. Dashboard Operational Real time: Hanya untuk Manajer dan Admin. Menampilkan informasi penting secara langsung di layar utama, tanpa perlu mengakses menu terpisah:
    - Total penjualan hari ini
    - Jumlah transaksi
    - Rata-rata nilai transaksi
    - Produk terlaris
    - Stok rendah
    - 5 transaksi terbaru
4. Riwayat Transaksi: Dapat dilihat oleh Manajer dan Admin dalam bentuk daftar transaksi lengkap dengan detail item dan total. Hanya admin yang dapat menghapus Riwayat.
5. Sistem Login 3 peran:
    - Kasir: hanya transaksi.
    - Manajer: transaksi + CRUD produk + dashboard + riwayat.

- Admin: semua fitur manajer + hapus riwayat

Solusi ini memungkinkan kafe menjalankan operasional harian dengan lebih rapi, cepat dan terkontrol tanpa kompleksitas fitur laporan eksternal.

2.2. Alur Kerja Sistem

Alur kerja sistem aplikasi POS dirancang secara linear dan intuitif, mengikuti siklus operasional toko harian. Berikut urutan fungsional utamanya:

1. Proses Login
   - User membuka aplikasi.
   - Sistem menampilkan layar login dengan input: username dan password.
   - Apabila tidak memiliki akun, dapat membuat akun baru.
   - Setelah login, pengguna diarahkan sesuai peran:
     - Kasir : langsung ke layar kasir.
     - Manager/Admin : ke dashboard utama.
2. Alur Dashboard (Manager/Admin)
   - Admin memilih menu "Dashboard".
   - Sistem menampilkan dashboard dari database yang berisi:
     - Today's Sale : Menampilkan total penjualan yang terjadi hari ini.
     - Transaction Today : Menampilkan jumlah transaksi yang terjadi hari ini.
     - Avg, Ticket : Menampilkan rata – rata penjualan.
     - Low Stock : Menampilkan barang yang stoknya tersisa sedikit.
     - Top Product : Menampilkan barang dengan penjualan terbanyak.
     - Recent History :Menampilkan transaksi terbaru.
3. Alur Transaksi Penjualan (Kasir/Manager/Admin)
   - Kasir memilih item dan menambahkannya ke keranjang.
   - Terapkan diskon (Opsional)
   - Saat checkout:
     - Sistem memeriksa ketersediaan stok tiap produk.
     - Jika stok cukup maka lanjut, jika tidak maka akan menampilkan peringatan.
   - Sistem menyimpan transaksi ke database, memperbarui stok, dan menampilkan struk digital.
4. Managemen Produk (Admin/Manager)
   - Akses menu "Kelola Produk"
   - Admin/Manager dapat menambah/edit/hapus produk
   - Data terbaru langsung tersedia di kasir
5. Riwayat (Manager/Admin)

- Dashboard muncul otomatis setelah login, menampilkan data real time
- Riwayat transaksi dapat dilihat dalam daftar
- Admin dapat menghapus entri riwayat

BAB III

IMPLEMENTASI SISTEM

3.1. Fitur Utama Aplikasi

1. Login dan Sign Up
   Fitur login dan signup menyediakan mekanisme autentikasi dan pendaftaran pengguna, di mana signup mengumpulkan data pengguna baru, melakukan validasi, lalu menyimpannya ke database, sementara login memverifikasi kredensial melalui UserDAO dan mengarahkan pengguna ke halaman sesuai perannya setelah berhasil.

2. Dashboard
   Fitur dashboard menampilkan ringkasan penting seperti produk paling banyak terjual, total transaksi, pendapatan harian, dan transaksi terbaru dengan mengambil data melalui DAO, sehingga manajer atau admin dapat langsung melihat kesehatan operasional toko.

3. Cashier
   Fitur cashier memungkinkan kasir menambahkan produk ke keranjang, menghitung subtotal, menerapkan diskon persentase, dan memproses pembayaran, sementara fungsi checkout menyimpan transaksi ke database melalui TransactionDAO serta memperbarui stok produk untuk menjaga konsistensi data.

4. Product Management
   Fitur create, read, update, delete product memungkinkan pengguna berwenang untuk membuat, membaca, memperbarui, dan menghapus data produk melalui ProductController yang berinteraksi dengan ProductDAO, termasuk fitur pencarian, pengisian otomatis form saat edit, validasi input, serta impor CSV agar manajemen produk lebih efisien.

5. History Transaction
   Fitur history menampilkan riwayat transaksi yang dapat difilter dan dicari, menampilkan detail tiap transaksi, dan menyediakan aksi tertentu seperti penghapusan transaksi bagi peran yang berhak, sehingga proses audit dan pelacakan transaksi dapat dilakukan dengan mudah dan aman.

6. Pembagian Peran
   Sistem peran mengatur batasan akses fitur di mana cashier hanya dapat menggunakan fitur kasir, manager dapat mengakses seluruh fungsi kecuali

penghapusan riwayat, sementara admin memiliki akses penuh termasuk hak untuk menghapus history sehingga kontrol dan keamanan sistem tetap terjaga.

3.2. Penjelasan Source Code

1. Controller

1) DashboardController.java

```
public void loadMetrics() {
    String todayPrefix = LocalDate.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));

    double revenue = transactionDAO.getDailyRevenue(todayPrefix);
    int txCount = transactionDAO.getTransactionsCountToday(todayPrefix);

    double avgTicket = txCount > 0 ? (revenue / txCount) : 0.0;

    dailyRevenueLabel.setText(String.format("Rp %.2f", revenue));
    transactionsCountLabel.setText(String.valueOf(txCount));
    avgTicketLabel.setText(String.format("Rp %.2f", avgTicket));

    // Top products
    List<ProductSales> top = transactionDAO.getTopProducts(todayPrefix, limit: 10);
    ObservableList<ProductSales> topList = FXCollections.observableArrayList(top);
    topProductsTable.setItems(topList);

    // Recent transactions
    List<Transaction> recent = transactionDAO.getRecentTransactions(limit: 10);
    ObservableList<Transaction> recentList = FXCollections.observableArrayList(recent);
    recentTransTable.setItems(recentList);
}
```

Mengatur logika untuk halaman dashboard setelah pengguna login. Controller ini menampilkan ringkasan seperti total produk, total transaksi, grafik penjualan, atau data penting lain yang diperlukan untuk overview aplikasi.

- initialize: Menyiapkan tampilan dashboard dan memanggil fungsi awal yang diperlukan.
- loadMetric: Mengambil metrik seperti jumlah produk, transaksi, atau pendapatan dan menampilkannya di dashboard.

2) HistoryController.java

```
private void handleDelete() {
    Transaction selected = historyTable.getSelectionModel().getSelectedItem();
    if (selected == null) {
        javafx.scene.control.Alert alert = new javafx.scene.control.Alert(javafx.scene.control.Alert.AlertType.INFORMATION);
        alert.setTitle(title: "No Selection");
        alert.setHeaderText(headerText: null);
        alert.setContentText(contentText: "Please select a transaction to delete.");
        alert.showAndWait();
        return;
    }

    javafx.scene.control.Alert confirm = new javafx.scene.control.Alert(javafx.scene.control.Alert.AlertType.CONFIRMATION);
    confirm.setTitle(title: "Confirm Delete");
    confirm.setHeaderText(headerText: null);
    confirm.setContentText("Are you sure you want to delete transaction ID " + selected.getId() + "? This will restore product stock.");
    java.util.Optional<javafx.scene.control.ButtonType> res = confirm.showAndWait();
    if (res.isPresent() && res.get() == javafx.scene.control.ButtonType.OK) {
        boolean ok = transactionDAO.deleteTransaction(selected.getId());
        if (ok) {
            javafx.scene.control.Alert info = new javafx.scene.control.Alert(javafx.scene.control.Alert.AlertType.INFORMATION);
            info.setTitle(title: "Deleted");
            info.setHeaderText(headerText: null);
            info.setContentText(contentText: "Transaction deleted successfully.");
            info.showAndWait();
            loadHistory();
        } else {
            javafx.scene.control.Alert err = new javafx.scene.control.Alert(javafx.scene.control.Alert.AlertType.ERROR);
            err.setTitle(title: "Error");
            err.setHeaderText(headerText: null);
            err.setContentText(contentText: "Failed to delete transaction. Check logs.");
            err.showAndWait();
        }
    }
}
```

Mengelola tampilan dan fungsi riwayat transaksi. Controller ini mengambil data transaksi yang sudah terjadi, memfilter tanggal, menampilkan detail transaksi, dan menyediakan fungsi pencarian riwayat.

- initialize: Menyiapkan tampilan history dan memuat data transaksi saat halaman dibuka.
- loadHistory: Mengambil seluruh transaksi dari database dan menampilkannya pada tabel.
- applyRolePermissions: Menyesuaikan fitur history yang dapat digunakan berdasarkan role user.
- handleDelete: Menghapus transaksi yang dipilih dan memperbarui tampilan history.

3) LoginController.java

```java
private void handleLogin() {
    String username = usernameField.getText();
    String password = passwordField.getText();

    User user = userDAO.validateUser(username, password);

    if (user != null) {
        try {
            // Set current user session
            App.setCurrentUser(user);
            // Navigate to main dashboard (MainController will adapt view based on role)
            App.setRoot(fxml: "main");
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        errorLabel.setText(value: "Invalid username or password");
        errorLabel.setVisible(true);
    }
}
```

Menangani proses login pengguna, termasuk validasi username/password, pengecekan ke database melalui UserDAO, serta pengalihan ke halaman utama bila login berhasil. Bertugas menjaga keamanan akses aplikasi.

- handleLogin: Memvalidasi kredensial pengguna dan mengarahkan ke dashboard jika berhasil.
- handleLogoutRedirect: Mengarahkan pengguna kembali ke halaman login saat logout dilakukan.
- handleSignUp: Membuka halaman pendaftaran akun baru untuk pengguna.

4) MainControlller.java

```
private void loadView(String fxml) {
    try {
        FXMLLoader loader = new FXMLLoader(App.class.getResource("/fxml/" + fxml + ".fxml"));
        Parent view = loader.load();
        contentArea.getChildren().clear();
        contentArea.getChildren().add(view);
    } catch (IOException e) {
        Logger.getLogger(MainController.class.getName()).log(Level.SEVERE, null, e);
    }
}
```

- initialize: Menyiapkan struktur utama tampilan aplikasi.
- applyRolePermissions: Mengatur menu dan fitur yang dapat diakses sesuai role pengguna.
- loadView: Memuat tampilan tertentu ke dalam area utama aplikasi.

5) PosController.java

```
@FXML
private void handleSearch() {
    String query = searchField.getText();
    if (query == null || query.isEmpty()) {
        loadProducts();
    } else {
        productList.setAll(productDAO.searchProducts(query));
        productTable.setItems(productList);
    }
}

@FXML
private void addToCart() {
    Product selected = productTable.getSelectionModel().getSelectedItem();
    if (selected != null) {
        if (selected.getStock() <= 0) {
            showAlert(title: "Error", message: "Out of stock!");
            return;
        }

        // Check if already in cart
        for (TransactionItem item : cartList) {
            if (item.getProductId() == selected.getId()) {
                if (item.getQuantity() < selected.getStock()) {
                    item.setQuantity(item.getQuantity() + 1);
                    cartTable.refresh();
                    updateTotal();
                } else {
                    showAlert(title: "Error", message: "Not enough stock!");
                }
                return;
            }
        }

        // Add new item
        TransactionItem newItem = new TransactionItem(id: 0, transactionId: 0, selected.getId(), quantity: 1, selected.getPrice());
        newItem.setProductName(selected.getName());
        cartList.add(newItem);
        updateTotal();
    }
}
```

Mengatur proses Point of Sale (kasir), seperti menambah item ke keranjang, menghitung total, membuat transaksi baru, mengurangi stok produk, serta melakukan penyimpanan transaksi lewat TransactionDAO.

- initialize: Menyiapkan tampilan POS dan memuat data awal seperti daftar produk.
- loadProducts: Mengambil seluruh produk dari database dan menampilkannya pada halaman POS.
- handleSearch: Menyaring produk berdasarkan kata kunci pencarian.
- addToCart: Menambahkan produk yang dipilih ke keranjang pembelian.
- removeFromCart: Menghapus item tertentu dari keranjang.
- updateTotal: Menghitung ulang total harga berdasarkan isi keranjang.
- handleCheckout: Menyimpan transaksi ke database dan mengosongkan keranjang setelah pembayaran.

- showAlert: Menampilkan pesan informasi atau peringatan kepada pengguna.

6) ProductController.java

```
private void loadProducts() {
    productList.setAll(productDAO.getAllProducts());
    productTable.setItems(productList);
}

private void populateForm(Product product) {
    codeField.setText(product.getCode());
    nameField.setText(product.getName());
    priceField.setText(String.valueOf(product.getPrice()));
    stockField.setText(String.valueOf(product.getStock()));
    categoryField.setText(product.getCategory());
}

@FXML
private void handleSave() {
    try {
        String code = codeField.getText();
        String name = nameField.getText();
        double price = Double.parseDouble(priceField.getText());
        int stock = Integer.parseInt(stockField.getText());
        String category = categoryField.getText();

        if (selectedProduct == null) {
            // Add new
            Product newProduct = new Product(id: 0, code, name, price, stock, category);
            productDAO.addProduct(newProduct);
        } else {
            // Update existing
            selectedProduct.setCode(code);
            selectedProduct.setName(name);
            selectedProduct.setPrice(price);
            selectedProduct.setStock(stock);
            selectedProduct.setCategory(category);
            productDAO.updateProduct(selectedProduct);
        }

        handleClear();
        loadProducts();
    } catch (NumberFormatException e) {
        showAlert(title: "Error", message: "Invalid input for price or stock.");
    }
}
```

Mengelola seluruh fungsi terkait produk, seperti menampilkan daftar produk, menambah produk baru, mengedit data produk, menghapus produk, dan menghubungkan UI dengan ProductDAO untuk operasi database.

- initialize: Mengatur tampilan awal halaman manajemen produk.
- loadProducts: Mengambil semua produk dari database untuk ditampilkan.
- populateForm: Mengisi form dengan data produk yang dipilih pengguna.
- handleSave: Menyimpan atau memperbarui produk melalui DAO.
- handleDelete: Menghapus produk berdasarkan pilihan pengguna.
- handleClear: Mengosongkan seluruh input pada form produk.
- handleSearch: Mencari produk berdasarkan kata kunci tertentu.
- handleImportCSV: Mengimpor data produk dari file CSV ke database.
- showAlert: Menampilkan pesan notifikasi atau kesalahan.

7) SignUpController.java

```java
private void handleSignUp() {
    String username = usernameField.getText().trim();
    String password = passwordField.getText();
    String confirmPassword = confirmPasswordField.getText();
    String role = roleComboBox.getValue();

    // Clear previous messages
    errorLabel.setVisible(false);
    successLabel.setVisible(false);

    // Validation
    if (username.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {
        errorLabel.setText(value: "All fields are required");
        errorLabel.setVisible(true);
        return;
    }

    if (username.length() < 3) {
        errorLabel.setText(value: "Username must be at least 3 characters");
        errorLabel.setVisible(true);
        return;
    }

    if (password.length() < 4) {
        errorLabel.setText(value: "Password must be at least 4 characters");
        errorLabel.setVisible(true);
        return;
    }

    if (!password.equals(confirmPassword)) {
        errorLabel.setText(value: "Passwords do not match");
        errorLabel.setVisible(true);
        return;
    }

    // Try to register user
    if (userDAO.registerUser(username, password, role)) {
        successLabel.setText(value: "Account created successfully! Redirecting to login...");
        successLabel.setVisible(true);

        // Redirect to login after 2 seconds
        javafx.animation.PauseTransition pause = new javafx.animation.PauseTransition(javafx.util.Duration.seconds(2));
        pause.setOnFinished(event -> {
            try {
                App.setRoot(fxml: "login");
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
        pause.play();
    } else {
        errorLabel.setText(value: "Username already exists. Please choose another.");
        errorLabel.setVisible(true);
    }
```

Menangani pendaftaran akun baru. Controller ini menerima input pengguna, melakukan validasi sederhana, lalu menyimpannya melalui UserDAO. Biasanya digunakan untuk menambah user admin/karyawan baru.

- initialize: Menyiapkan tampilan awal halaman pendaftaran akun.
- handleSignUp: Memproses registrasi akun baru dan menyimpannya di database.
- handleBackToLogin: Mengembalikan pengguna ke halaman login.

2. DAO

1) ProductDAO.java

```java
public List<Product> getAllProducts() {
    List<Product> products = new ArrayList<>();
    String sql = "SELECT * FROM products";

    try (Connection conn = DatabaseHelper.connect();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            products.add(new Product(
                    rs.getInt("id"),
                    rs.getString("code"),
                    rs.getString("name"),
                    rs.getDouble("price"),
                    rs.getInt("stock"),
                    rs.getString("category")
            ));
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    return products;
}

public void addProduct(Product product) {
    String sql = "INSERT INTO products(code, name, price, stock, category) VALUES(?,?,?,?,?)";

    try (Connection conn = DatabaseHelper.connect();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, product.getCode());
        pstmt.setString(2, product.getName());
        pstmt.setDouble(3, product.getPrice());
        pstmt.setInt(4, product.getStock());
        pstmt.setString(5, product.getCategory());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}
```

Bertugas melakukan semua operasi database untuk produk, seperti INSERT, UPDATE, DELETE, dan SELECT. DAO ini memisahkan logika database dari controller agar aplikasi lebih rapi dan mudah dirawat.

- addProduct: Menambahkan produk baru ke database.
- updateProduct: Memperbarui data produk yang sudah ada.
- deleteProduct: Menghapus produk tertentu berdasarkan ID.
- searchProducts: Mencari produk berdasarkan kata kunci.
- getAllProducts: Mengambil seluruh data produk dari database.

2) TransactionDAO.java

```java
public class TransactionDAO {

    public void saveTransaction(Transaction transaction) {
        String sqlTrans = "INSERT INTO transactions(date, total, discount, cashier_name) VALUES(?,?,?,?)";
        String sqlItem = "INSERT INTO transaction_items(transaction_id, product_id, quantity, price_at_sale) VALUES(?,?,?,?)";
        String sqlUpdateStock = "UPDATE products SET stock = stock - ? WHERE id = ?";

        Connection conn = null;
        try {
            conn = DatabaseHelper.connect();
            conn.setAutoCommit(false); // Start transaction

            // Insert Transaction
            int transId = -1;
            try (PreparedStatement pstmt = conn.prepareStatement(sqlTrans, Statement.RETURN_GENERATED_KEYS)) {
                pstmt.setString(1, transaction.getDate());
                    pstmt.setDouble(2, transaction.getTotal());
                    pstmt.setDouble(3, transaction.getDiscount());
                    pstmt.setString(4, transaction.getCashierName());
                pstmt.executeUpdate();

                try (ResultSet rs = pstmt.getGeneratedKeys()) {
                    if (rs.next()) {
                        transId = rs.getInt(1);
                    }
                }
            }

            if (transId != -1) {
                // Insert Items and Update Stock
                try (PreparedStatement pstmtItem = conn.prepareStatement(sqlItem);
                     PreparedStatement pstmtStock = conn.prepareStatement(sqlUpdateStock)) {

                    for (TransactionItem item : transaction.getItems()) {
                        // Insert Item
                        pstmtItem.setInt(1, transId);
                        pstmtItem.setInt(2, item.getProductId());
                        pstmtItem.setInt(3, item.getQuantity());
                        pstmtItem.setDouble(4, item.getPriceAtSale());
                        pstmtItem.addBatch();

                        // Update Stock
                        pstmtStock.setInt(1, item.getQuantity());
                        pstmtStock.setInt(2, item.getProductId());
                        pstmtStock.addBatch();
                    }
                    pstmtItem.executeBatch();
                    pstmtStock.executeBatch();
                }
            }
```

Digunakan untuk menyimpan transaksi dan detailnya ke database. DAO ini bertanggung jawab membuat record transaksi baru, menyimpan item transaksi satu per satu, dan mengambil data transaksi untuk history.

- saveTransaction: Menyimpan transaksi dan seluruh itemnya ke database.
- getAllTransactions: Mengambil seluruh data riwayat transaksi.
- getDailyRevenue: Mengambil total pendapatan pada hari berjalan.
- getTransactionsCountToday: Mengambil jumlah transaksi yang terjadi hari ini.
- getTopProducts: Mengambil daftar produk dengan penjualan terbanyak.
- getRecentTransactions: Mengambil transaksi terbaru untuk ditampilkan di dashboard.
- deleteTransaction: Menghapus transaksi beserta itemnya dari database.

3) UserDAO.java

```java
public User validateUser(String username, String password) {
    String sql = "SELECT * FROM users WHERE username = ? AND password = ?";

    try (Connection conn = DatabaseHelper.connect();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, username);
        pstmt.setString(2, password);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            return new User(
                    rs.getString("username"),
                    rs.getString("password"),
                    rs.getString("role")
            );
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    return null;
}

public boolean userExists(String username) {
    String sql = "SELECT * FROM users WHERE username = ?";

    try (Connection conn = DatabaseHelper.connect();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, username);
        ResultSet rs = pstmt.executeQuery();
        return rs.next();
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    return false;
}

public boolean registerUser(String username, String password, String role) {
    // Check if user already exists
    if (userExists(username)) {
        return false;
    }

    String sql = "INSERT INTO users (username, password, role) VALUES (?, ?, ?)";

    try (Connection conn = DatabaseHelper.connect();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, username);
        pstmt.setString(2, password);
        pstmt.setString(3, role != null ? role : "cashier");
        pstmt.executeUpdate();
        return true;
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    return false;
}
```

Mengatur proses akses data untuk akun pengguna, seperti login (cek username/password), menambah user baru, mencari user berdasarkan ID, atau mengambil daftar user yang ada.

- validateUser: Memeriksa kecocokan username dan password saat login.
- userExists: Mengecek apakah username yang ingin didaftarkan sudah digunakan.
- registerUser: Menyimpan akun baru ke dalam database.

3. Model

1) Product.java

```java
public class Product {
    private int id;
    private String code;
    private String name;
    private double price;
    private int stock;
    private String category;

    public Product(int id, String code, String name, double price, int stock, String category) {
        this.id = id;
        this.code = code;
        this.name = name;
        this.price = price;
        this.stock = stock;
        this.category = category;
    }

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }

    public int getStock() { return stock; }
    public void setStock(int stock) { this.stock = stock; }

    public String getCategory() { return category; }
    public void setCategory(String category) { this.category = category; }
}
```

Model yang merepresentasikan satu produk. Berisi atribut seperti id, nama, harga, dan stok. Dipakai untuk menampung data produk yang diambil dari database atau dikirim ke DAO.

2) ProductSales.java

```java
package com.minipos.model;

public class ProductSales {
    private String name;
    private int quantity;
    private double revenue;

    public ProductSales(String name, int quantity, double revenue) {
        this.name = name;
        this.quantity = quantity;
        this.revenue = revenue;
    }

    public String getName() { return name; }
    public int getQuantity() { return quantity; }
    public double getRevenue() { return revenue; }
}
```

Model khusus untuk laporan penjualan produk. Biasanya berisi nama produk, jumlah terjual, total pendapatan, dan data yang digunakan untuk grafik atau laporan penjualan.

3) Transaction.java

```java
package com.minipos.model;

import java.util.List;

public class Transaction {
    private int id;
    private String date;
    private double total;
    private String cashierName;
    private double discount;
    private List<TransactionItem> items;

    public Transaction(int id, String date, double total, String cashierName) {
        this(id, date, total, discount: 0.0, cashierName);
    }

    public Transaction(int id, String date, double total, double discount, String cashierName) {
        this.id = id;
        this.date = date;
        this.total = total;
        this.discount = discount;
        this.cashierName = cashierName;
    }

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getDate() { return date; }
    public void setDate(String date) { this.date = date; }

    public double getTotal() { return total; }
    public void setTotal(double total) { this.total = total; }

    public double getDiscount() { return discount; }
    public void setDiscount(double discount) { this.discount = discount; }

    public String getCashierName() { return cashierName; }
    public void setCashierName(String cashierName) { this.cashierName = cashierName; }

    public List<TransactionItem> getItems() { return items; }
    public void setItems(List<TransactionItem> items) { this.items = items; }
}
```

Model untuk satu transaksi. Menyimpan informasi seperti id transaksi, tanggal, total harga, user yang melakukan, dan daftar item transaksi. Ini representasi utama dalam proses POS.

4) TransactionItem.java

```java
package com.minipos.model;

public class TransactionItem {
    private int id;
    private int transactionId;
    private int productId;
    private int quantity;
    private double priceAtSale;
    private String productName; // For display convenience

    public TransactionItem(int id, int transactionId, int productId, int quantity, double priceAtSale) {
        this.id = id;
        this.transactionId = transactionId;
        this.productId = productId;
        this.quantity = quantity;
        this.priceAtSale = priceAtSale;
    }

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public int getTransactionId() { return transactionId; }
    public void setTransactionId(int transactionId) { this.transactionId = transactionId; }

    public int getProductId() { return productId; }
    public void setProductId(int productId) { this.productId = productId; }

    public int getQuantity() { return quantity; }
    public void setQuantity(int quantity) { this.quantity = quantity; }

    public double getPriceAtSale() { return priceAtSale; }
    public void setPriceAtSale(double priceAtSale) { this.priceAtSale = priceAtSale; }

    public String getProductName() { return productName; }
    public void setProductName(String productName) { this.productName = productName; }
}
```

Mewakili satu item dalam transaksi. Berisi id produk, kuantitas, harga, dan subtotal. Digunakan untuk mencatat produk apa saja yang dibeli di satu transaksi.

5) User.java

```java
public class User {
    private String username;
    private String password;
    private String role;

    public User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }
}
```

Model untuk menyimpan data pengguna aplikasi, seperti id, username, password (biasanya sudah di-hash), serta role (admin/kasir). Dipakai saat autentikasi dan manajemen user.

4. Util

1) DatabaseHelper.java

```java
public class DatabaseHelper {
    private static final String DB_URL = "jdbc:sqlite:minipos.db";

    public static Connection connect() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(DB_URL);
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
        return conn;
    }

    public static void initializeDatabase() {
        try (Connection conn = connect();
             Statement stmt = conn.createStatement()) {

            // Users table
            String sqlUsers = "CREATE TABLE IF NOT EXISTS users (" +
                    "username TEXT PRIMARY KEY," +
                    "password TEXT NOT NULL," +
                    "role TEXT NOT NULL" +
                    ");";
            stmt.execute(sqlUsers);

            // Products table
            String sqlProducts = "CREATE TABLE IF NOT EXISTS products (" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                    "code TEXT UNIQUE NOT NULL," +
                    "name TEXT NOT NULL," +
                    "price REAL NOT NULL," +
                    "stock INTEGER NOT NULL," +
                    "category TEXT" +
                    ");";
            stmt.execute(sqlProducts);

            // Transactions table
            String sqlTransactions = "CREATE TABLE IF NOT EXISTS transactions (" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                    "date TEXT NOT NULL," +
                    "total REAL NOT NULL," +
                    "discount REAL DEFAULT 0," +
                    "cashier_name TEXT" +
                    ");";
            stmt.execute(sqlTransactions);

            // Transaction Items table
            String sqlTransactionItems = "CREATE TABLE IF NOT EXISTS transaction_items (" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                    "transaction_id INTEGER NOT NULL," +
                    "product_id INTEGER NOT NULL," +
                    "quantity INTEGER NOT NULL," +
                    "price_at_sale REAL NOT NULL," +
                    "FOREIGN KEY(transaction_id) REFERENCES transactions(id)," +
                    "FOREIGN KEY(product_id) REFERENCES products(id)" +
                    ");";
```

Kelas utilitas untuk mengatur koneksi ke database. Menyediakan fungsi untuk membuka dan menutup koneksi, mengelola driver JDBC, serta membantu DAO agar tidak perlu mengulang kode koneksi.

- initializeDatabase: Menyediakan koneksi database dan memastikan tabel siap digunakan.

BAB IV

ANALISIS

4.1. Tantangan

Dalam pengembangan program Point of Sale (POS) untuk Cafe, sejumlah tantangan teknis dan desain muncul sejak tahap perencanaan hingga implementasi. Salah satu tantangan utama adalah mengintegrasikan berbagai modul seperti transaksi kasir, manajemen produk, pembaruan stok real time, sistem login berbasis tiga peran (Kasir, Manajer, Admin) dan dashboard operasional ke dalam satu sistem yang ringan namun tetap stabil. Selain itu, memastikan konsistensi data antara antarmuka pengguna, logika bisnis dan basis data SQL memerlukan perencanaan struktur yang matang, terutama dalam hal validasi input, penanganan error saat transaksi serta pencegahan stok negatif. Tantangan lain meliputi pembatasan fitur agar aplikasi tetap sederhana dan tidak membebani perangkat, misalnya memfokuskan dashboard hanya sebagai tampilan real time. Hal ini menuntut tim untuk membuat keputusan desain yang tepat antara fungsionalitas dan kesederhanaan, sekaligus memastikan pengalaman pengguna tetap intuitif meski tanpa fitur yang sering dianggap "standar" pada sistem POS komersial.

4.2. Peluang

Di balik tantangan tersebut, pengembangan program POS justru membuka peluang besar untuk penerapan konsep ilmu komputer secara nyata dan terpadu. Program ini menjadi wadah ideal untuk menerapkan prinsip pemrograman berorientasi objek (OOP) melalui pembuatan kelas yang terstruktur seperti Product, Transaction, User, dan Dashboard serta pemanfaatan enkapsulasi, pewarisan dan polymorphism dalam arsitektur aplikasi. Selain itu, penggunaan SQL sebagai database lokal memungkinkan eksplorasi praktis dalam manajemen data, query dasar serta integrasi antara Java dan database tanpa dependensi server. Dari sisi desain sistem, penerapan role based access control (RBAC) dengan tiga peran berbeda melatih pemahaman tentang keamanan aplikasi dan pembatasan akses berbasis hak pengguna. Yang tak kalah penting, program ini menghasilkan solusi digital yang benar-benar dapat digunakan oleh pelaku usaha mikro, sehingga memberikan dampak nyata dalam meningkatkan akurasi pencatatan, efisiensi operasional dan profesionalisme usaha. Dengan demikian, meskipun sederhana, aplikasi POS bukan hanya proyek akademis, tetapi juga langkah awal menuju pengembangan perangkat lunak yang human centered, andal dan berorientasi pada kebutuhan pengguna riil.

BAB V

PENUTUP

5.1. Kesimpulan

Pembuatan aplikasi Point of Sale (POS) untuk Cafe telah berhasil menghadirkan solusi digital yang ringan, fokus, dan sesuai kebutuhan operasional dasar. Aplikasi ini menjawab permasalahan umum seperti pencatatan manual, ketidakakuratan stok dan kurangnya pemantauan penjualan harian tanpa membebani pengguna dengan fitur kompleks seperti laporan eksternal.

Dengan fitur inti seperti transaksi kasir, manajemen produk, dashboard real time dan sistem tiga peran (Kasir, Manajer, Admin), aplikasi ini mampu meningkatkan akurasi data, mempercepat proses layanan, dan memberikan gambaran operasional harian secara langsung. Penggunaan SQLite memastikan aplikasi dapat dijalankan di perangkat biasa tanpa internet atau server.

Secara keseluruhan, POS bukan hanya alat bantu transaksi, tetapi juga sistem pengelolaan toko yang terstruktur dan aman. Dengan menghilangkan fitur laporan eksternal, aplikasi justru menjadi lebih sederhana, fokus pada kebutuhan inti dan mudah dioperasikan oleh pemilik usaha mikro yang tidak memiliki latar belakang teknis..